(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 July 2003 (03.07.2003)

PCT

(10) International Publication Number
WO 03/053531 A1

(51) International Patent Classification[7]: A63F 9/24

(21) International Application Number: PCT/IL02/01000

(22) International Filing Date:
11 December 2002 (11.12.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/338,674 11 December 2001 (11.12.2001) US

(71) Applicant *(for all designated States except US)*: **MAJOREM LTD.** [IL/IL]; Lechi 31 (Box 18), 5 1200 Bnei Brak (IL).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **AMIR, Gideon** [IL/IL]; Kehilat Zion 46, 46 382 Herzlia (IL). **AXELROD, Ramon** [IL/IL]; Kdoshey Kahir 38, 58 363 Holon (IL). **MENDELSON, Yuval** [IL/IL]; Nordau Street 45, 46 585 Herzlia (IL).

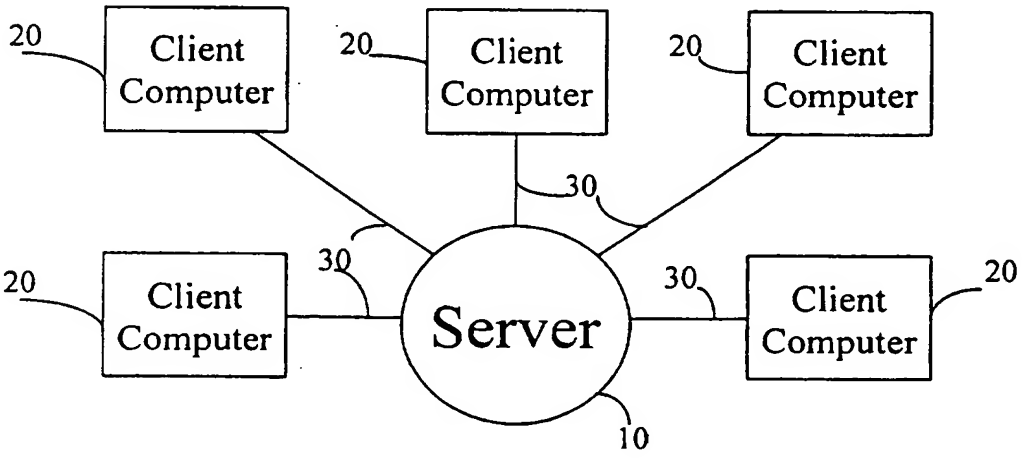
(74) Agent: **G. E. EHRLICH (1995) LTD.**; Bezalel Street 28, 52 521 Ramat Gan (IL).

(81) Designated States *(national)*: AE, AG, AL, AM, AT (utility model), AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ (utility model), CZ, DE (utility model), DE, DK (utility model), DK, DM, DZ, EC, EE (utility model), EE, ES, FI (utility model), FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK (utility model), SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: MASSIVE MULTIPLAYER REAL-TIME PERSISTANT NETWORK GAME ENGINE



(57) Abstract: A system for improving the operation of network based games providing reduced bandwidth requirement and increased persistency of the game. In one embodiment the invention provides for reduced bandwidth by providing each client computer (20) with updates regarding only the inner and border regions associated with the specific client computer. Information regarding the balance of the game space is only transferred as required. In another embodiment the system provides for increased persistency by allowing playing pieces to be transferred from one player to another player. In a preferred embodiment the playing pieces are transferred from a departing player to an artificial agent to maintain the player. In yet another embodiment the invention provides for reduced bandwidth by guaranteeing the data path (30) from the server (10) to the client computer (20), while allowing for failed communication in the reverse path.

WO 03/053531 A1

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

MASSIVE MULTIPLAYER REAL-TIME PERSISTENT NETWORK GAME ENGINE

BACKGROUND OF THE INVENTION

The invention relates generally to the field of network game playing, and more specifically to an engine for enabling real time game playing over a network.

With the rapid increase in the number of people worldwide who have access to network connected computing devices such as computers, cellular phones and personal digital assistants, there is an ever increasing demand for network based gameplay. In particular the rapid growth of the Internet, with it global reach, has spurred the interest in networked gameplay.

Network games are often classified according to the theme or genre of the game. Current popular genres include role playing games, educational games, adventure games, arcade or action games, simulation games and strategy games, which in a network environment can be played with other players. In some games, most notably strategy games, the player controls multiple individual game elements that are moved or manipulated in a virtual game world or map.

Network games are further classified according to the time frames in which players interact with one another. In turn-based games each player is typically given a defined period of time to make decisions and input instructions for the next move. The state of the game and all pieces are thus updated in lock step, typically on the order of minutes.

In real-time network games, the state of the game is updated based on moves of active players, with no time pre-ordained time constraint. Thus the state of the game is in constant flux, requiring frequent updating. The bandwidth and processing needs for a real-time network game is thus a function of the number of active players and the number of game elements. In prior art systems, the location of all game elements, or at least all moved game elements are broadcast to all active players on a regular basis.

In a typical multiple player environment, knowledge of the position of all the game elements of all the relevant players in a particular area of interest is needed for the user to make an informed choice regarding his move. As the number of players increase,

and the number of individual game elements controlled by each player increases, the need for updating the position of all the game elements of all the relevant players demands increased communication bandwidth and processing power. This need is further amplified in a real-time network game, in which moves of elements by any player may occur at any time.

An additional aspect of network-based games is the level of persistency, and two types of games may be defined, session-based and persistent. In a session-based game, the game lasts only as long as all the players remain connected. The loss of a single player thus ends the game. In a persistent game, the game remains active even when some of the players disconnect. In current games of the persistent type, when a player of a network-based game decides to leave the game, the player's game elements are typically left unavailable to other players. Furthermore, the leaving player is "left behind" as the game continues in the player's absence. Only when the player reconnects can the player's game elements be moved.

Thus there is a need for a system that will allow players leaving a persistent game to enable their game elements to be actively involved in the game even while the player is not connected. Furthermore there is a need to minimize the bandwidth involved in updating the position of multiple game elements in a network-based game.

SUMMARY OF THE INVENTION

Accordingly, it is a principal object of the present invention to overcome the disadvantages of prior art network game playing. This is provided in the present invention by a method for network gaming involving a virtual game area divisible into regions, and a plurality of networked players each having at least one playing piece, comprising: determining for each region and for each player whether the region is an inner region in which the player has a game piece, a border region which is within access of a player's piece or an outer region which is not accessible in this move by the player; maintaining a relationship of players and game elements, the relationship maintaining whether said region is an inner region or a border region for each player; and notifying only those players for which the region is a border region of changes in the game state.

In a preferred embodiment the invention further provides for notifying only those players for which the region is an inner region of all player actions in said inner region. In another preferred embodiment the invention further provides for notifying only those players for which the region is an inner region or a border region of all player actions in those regions. In one embodiment the regions are squares, and in another embodiment the regions are hexagons.

In an exemplary embodiment the invention further provides for game persistency by: receiving notification by a server of an impending disconnection of a player; searching available computing resources to determine an available substitute playing agent; choosing the available substitute playing agent; and updating the maintained relationship with substitute playing agent. In one further preferred embodiment the invention provides for the searching to comprise finding the available substitute agent having inner regions in common with the disconnecting player. In another further preferred embodiment the invention provides for the searching to comprise finding the available substitute agent having the maximum amount of inner regions in common with the disconnecting player. In yet another further preferred embodiment the invention provides for the available substitute agent to be an artificial agent.

The invention also provides for a system for network gaming involving a virtual game area divisible into regions comprising: a plurality of networked players, each player having at least on playing piece; a server, connected to each playing entity by a data connection, the server providing: determining functionality for determining for each region and for each player whether the region is an inner region in which the player has a game piece, a border region which is within access of a players piece or an outer region which is not accessible in this move by the player; maintaining functionality associated with the determining functionality for maintaining in the server a relationship between players and game elements, the relationship comprising whether the region is an inner region or a border region for each player; and notifying functionality associated with the maintaining functionality for notifying only those players for which the region is a border region of changes in the game state.

In one preferred embodiment the invention further provides for the notifying functionality to notify only those players for which the region is an inner region of player

actions in the inner region. In another preferred embodiment the invention further provides the notifying functionality to notify only those players for which the region is an inner region or a border region of player actions in those regions. In one embodiment the regions are squares, and in another embodiment the regions are hexagons.

In another preferred embodiment the invention further provides: receiving functionality for receiving notification by the server of an impending disconnection of a player; searching functionality associated with the receiving functionality for an available substitute playing agent; bandwidth availability functionality for determining the currently available bandwidth to each available substitute playing agent; computing resource availability functionality for determining the currently available computing resources of each available substitute playing agent; choosing functionality associated with the searching functionality the bandwidth availability functionality and the computing resource availability functionality for choosing the available substitute playing agent having required bandwidth requirements and available computing resources; updating functionality associated with the choosing functionality and the maintaining functionality for updating the maintaining functionality with the substitute available playing agent.

In one further preferred embodiment the invention provides for the choosing functionality to find the available substitute agent having inner regions in common with the disconnecting player. In another further preferred embodiment the invention provides for the choosing functionality to find the available substitute agent having the maximum amount of inner regions in common with the disconnecting player. In one further preferred embodiment the available playing agent is an artificial agent.

The invention also provides for a method for persistent network gaming for a plurality of networked players, comprising; receiving notification by a server of an impending disconnection of a networked player, the player having at least one gaming piece; searching available computing resources to determine an available substitute playing agent; choosing the available substitute playing agent; and transferring control of the gaming pieces to the available agent.

In one further preferred embodiment the invention provides for the searching to further comprise finding the available substitute agent having minimum bandwidth

requirements. In another further preferred embodiment the available substitute agent is an artificial agent. In another further preferred embodiment at least some of said available computing resources are co-resident with at least one networked player.

The invention also provides for a system for persistent network gaming comprising: a plurality of networked players; and a server providing: receiving functionality for receiving notification by the server of an impending disconnection of a player; searching functionality associated with the receiving functionality for searching available computing resources to determine an available substitute playing agent; bandwidth determining functionality associated with the searching functionality for determining the bandwidth requirements and availability for each available substitute playing agent; choosing functionality associated with the ranking functionality for choosing the available substitute playing agent with minimum bandwidth requirements; transferring functionality associated with the choosing functionality for transferring at least one game piece to the chosen available substitute playing agent.

In one further preferred embodiment the available playing agent is an artificial agent. In another further preferred embodiment at least some of the additional computing resources are co-resident with at least one networked player.

The invention also provides for a system for network based gaming comprising: a client computer; and a data connection connecting the client computer to a server; the client computer comprising an application program and a communication module, the communication module further comprising: assembling functionality for assembling messages from the application program into packets for delivery to the server computer over the data connection; message storage functionality associated with the assembling functionality for storing a copy of the messages prior to said delivery; received packet storage number functionality for maintaining a list of received packets from said server; disassembler functionality associated with the received packet storage number functionality for disassembling the received packets from the server, the disassembler functionality further providing: examining functionality associated with the message storage functionality for examining the received packet for an acknowledgement or retransmit request of the stored message by the server; marking functionality associated with the message storage functionality and the examination functionality for marking the

stored messages with the acknowledgement or retransmit request ; and transfer functionality associated with the marking functionality for transferring the marked messages from the message storage to the application program.

In one preferred embodiment the invention further provides for the disassembler functionality to further provide calculating functionality associated with the storage functionality for determining if the received packet is in numerical sequence with previously received packets. In another preferred embodiment the invention further provides for the disassembler functionality to further provide notifying functionality associated with the calculating functionality for generating a retransmit request to the server.

The invention also provides for a method for network gaming comprising: assembling messages from an application program into a packet for delivery to a server computer over a data connection; storing a copy of the messages; disassembling a received packet from the server computer comprising: storing a serial number of the received packet; examining the received packet for successful acknowledgement or retransmit request of at least one stored message; marking the stored messages with the result of the examination; and transferring the marked stored messages from the message storage to the application program.

In a preferred embodiment the invention also provides for calculating if said received packet is in numerical sequence with previously received packets. Still further preferably the invention provides for generating a retransmit request in an out of sequence event. In another preferred embodiment the invention further provides for the application program acting upon the acknowledged messages, and generating new messages in response to said unacknowledged messages.

Additional features and advantages of the invention will become apparent from the following drawings and description.

BRIEF DESCRIPTION OF THE DRAWINGS

For a better understanding of the invention and to show how the same may be carried into effect, reference will now be made, purely by way of example, to the accompanying drawings.

With specific reference now to the drawings in detail, it is stressed that the particulars shown are by way of example and for purposes of illustrative discussion of the preferred embodiments of the present invention only, and are presented in the cause of providing what is believed to be the most useful and readily understood description of the principles and conceptual aspects of the invention. In this regard, no attempt is made to show structural details of the invention in more detail than is necessary for a fundamental understanding of the invention, the description taken with the drawings making apparent to those skilled in the art how the several forms of the invention may be embodied in practice. In the accompanying drawings:

Fig. 1 illustrates a high level block diagram of a system utilizing the teaching of the invention;

Fig. 2 illustrates a high level flow block diagram of a client computer according to the teaching of the invention;

Fig. 3 illustrates a high level flow chart of the operation of the disassembler of Fig. 2 according to the teaching of the invention;

Fig. 4 illustrates a high level functional block diagram of a system according to the teaching of the invention;

Fig. 5 illustrates a high level block diagram of a server according to the teaching of the invention;

Fig. 6 illustrates a high level flow chart of the operation of the disassembler of Fig. 5 according to the teaching of the invention;

Fig. 7a illustrates a graphical representation of a reverse position structure on a first client computer;

Fig. 7b illustrates a graphical representation of a reverse position structure of a second client computer;

Fig. 8 illustrates a graphical representation of the structure of a single region of a reverse position structure on the server;

Fig. 9 illustrates a high level block diagram of network based game comprising playing agents;

Fig. 10 illustrates a high level flow chart of the operation of server 10; and

Fig. 11 illustrates a high level functional block diagram of the operation of server 10.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present embodiments provide a system and method enabling massive multiplayer real time network gaming. A preferred embodiment allows for reduced bandwidth by maintaining partitioning of the game space into regions. Regions in which a player has active pieces are defined as inner regions, and the player can calculate interactions with other players in an inner region. Regions adjacent to an inner region are defined as border regions, and the server broadcasts to players moves of all game elements in the player's inner and border regions. For each region the server keeps track of all game pieces and for which players the region is an inner or border region. Information regarding changes in the game state in that region is only transmitted to players for which the region is defined as a border inner region.

Another preferred embodiment allows transfer of control of game elements by a player to another playing entity. Preferably the other playing entity is an artificial playing entity operable to maintain the player's game elements in the game.

Another preferred embodiment allows for reduced bandwidth requirements by allowing for an asymmetrical protocol between a client computer and a server, in which delivery of messages from the server to the client computer are guaranteed, but messages from the client computer to the server are not.

Before explaining at least one embodiment of the invention in detail, it is to be understood that the invention is not limited in its application to the details of construction and the arrangement of the components set forth in the following description or illustrated in the drawings. The invention is applicable to other embodiments or of being practiced or carried out in various ways. Also, it is to be understood that the phraseology and terminology employed herein is for the purpose of description and should not be regarded as limiting.

Fig. 1 illustrates a high level block diagram of a system according to a preferred embodiment operative in accordance with the present invention. The system comprises a server 10, multiple client computers 20 and data connections 30. Server 10 is connected

to each of the multiple client computers 20 by data connection 30. In a preferred embodiment data connection 30 comprises a UDP/IP connection over a packet based network and optionally the packet based network is the Internet. A game control unit (not shown) located on server 10 is operative to control a network-based game, whose players interact via an application (not shown) running on client computers 20.

Fig. 2 illustrates a high level block diagram of a client computer 20 of Fig. 1 operative in accordance with a preferred embodiment of the present invention. Client computer 20 comprises application 40 and communication module 50. Communication module 50 comprises transmit message buffer 60, packet assembler 70, message storage 80, transmit packet buffer 90, transmission agent 100, one end of data connection 30, receive packet buffer 110, packet disassembler 120, received message buffer 130 and received message number storage 140. Application 40 is associated with transmit message buffer 60 at the input to communication module 50, and the output transmit message buffer 60 is associated with an input of packet assembler 70. A first output of packet assembler 70 is associated with the input of message storage 80, and a second output of packet assembler 70 is associated with the input of packet transmit buffer 90. The output of packet transmit buffer 90 is associated with an input of transmission agent 100, and an output of transmission agent 100 is associated with the input of receive packet buffer 110. Transmission agent 90 is further associated bi-directionally to one end of data connection 30. The output of receive packet buffer 110 is associated with an input of packet disassembler 120, and a second input of packet disassembler 120 is associated with the output of message storage 80. One output of packet disassembler 120 is associated with the input of received message buffer 130, and the output of received message buffer 130 is associated, across the output of communicant module 50, with an input of application 40. Packet disassembler 120 is further associated bi-directionally with received message number storage 140, and the output of received message number storage 140 is associated with an input of packet assembler 70. A further output of packet disassembler 120 is associated with an input of packet assembler 70.

In operation, application 40 generates messages for transmittal to server 10. The messages may be queries on the state of the game, typically for identification regarding game elements and players in a particular area of the game, requests to move one or more

game elements controlled by the application, thus changing the game state, or other management messages. Preferably application 40 does not act upon its request to change the game state, and instead functions in a manner to be describer further below. Messages generated by application 40 are stored in transmit message buffer 60, which in a preferred embodiment comprises a first in, first out (FIFO) memory. In a preferred embodiment the messages are sequentially enumerated for enabling individual acknowledgement. Packet assembler 70 operates at regular intervals to collect messages from transmit message buffer 60, and to assemble them into packets. In one embodiment, packet assembler 70 operates once every 200 milliseconds, and in another embodiment packet assembler 70 operates once every 300 milliseconds. Optionally, packet assembler 70 further operates to sequentially number the packets being assembled. The packets are assembled with a header, which in a preferred embodiment comprises the serial number of the first message in the packet, the number of messages in the packet and the number of the last packet received from the server, which is collected from received message number storage 140. Optionally, the header further comprises the sequential number of the packet.

In the event message storage 60 has no messages to transmit, an empty message is created indicating client computer 20 and application 40 are still actively connected to server 10 (not shown) for the purpose of playing the network-based game. A copy of the messages sent in the packet are stored in message storage 80, and the packet is sent to transmit packet buffer 90. Transmission agent 100 operates to transmit any packets found in transmit packet buffer 90 over data connection 30 to server 10 (not shown). In a preferred embodiment transmission agent 100 operates under the UDP/IP protocol.

Packets received by transmission agent 100 from server 10 (not shown) over data connection 30 are transferred to received packet buffer 110. Packet disassembler 120 operates promptly to retrieve the received packet from received packet buffer 110 and disassemble the packet into a header and messages destined for application 40. The header comprises the sequential number of any acknowledged messages received by server 10 (not shown) from client computer 20, or a request to retransmit specific messages, or multiple messages. The header further comprises a serial number of the current packet. The serial number of the current packet received from the server is stored

in received message number storage 140. The sequential number of the acknowledged message received is compared with the sequential number of the messages stored in message storage 80, and the messages with acknowledged sequential number are transferred to received message buffer 130, and a flag or header is appended indicating that they are messages generated by application 40 that have been acknowledged. The message is thus removed from message storage 80. In the event that a retransmit request is received, the appropriate messages are retrieved from message storage 80, and the messages are transferred to received message buffer 130, and a flag or header is appended indicating that they are messages generated by application 40 that have not been acknowledged. Application 40 generates new messages, which may be different than the previously stored messages due to changes in the game state, or changes in the users intent.

The serial number of the packet received is inspected to ensure that it is in sequence with the previously received packet, as stored in received message number storage 140. In the event that a packet is missing, packet disassembler 120 notifies packet assembler 70 to generate a retransmit request. Packet assembler 70 utilizes the last packet number in sequence found in message number storage 140 in the retransmit request, thus indicating the last known proper received packet. In the event that no packet is missing, the messages are extracted from the packet, and sent to received message buffer 130. Application 40 polls received message buffer 130 to accept any new received messages.

Fig. 3 illustrates a high level flow chart of the operation of disassembler 120 of Fig. 2. In step 1000, the disassembler receives a packet from receive packet buffer 110, and in step 1010 the serial number of the received packet is identified and stored in received message number storage 140. In step 1020 the header of the received packet is examined to determine if comprises any retransmit requests. In the event that it comprises any a retransmit requests, in step 1030 the messages are transferred from message storage 80 and in step 1040 the messages are marked to indicate they have not been acknowledged. In step 1050 the marked messages are sent to the application 40 through received message buffer 130.

In the event that in step 1020 the header of the received packet was determined not to comprise a retransmit request, in step 1060 the serial number of the packet is checked against previously stored serial numbers in received message number storage 140, to determine if the packet has been received in sequence. In the event that a packet is missing from the sequence, in step 1070 the packet assembler 70 is notified to send a retransmit request to the server, otherwise in step 1080 the packet is examined to find message acknowledgements. In the event that the received packet comprises an acknowledgement, in step 1090 the stored message with matching sequential number is transferred from message storage 80 and in step 1100 the messages are marked to indicate they have been acknowledged. In step 1110 the messages are sent to application 40 through received message buffer 130.

It will be noted that messages sent from the server are checked to ensure that they have been fully received, and application 40 will receive no further messages until any missing messages are received. This ensures that any game state messages sent by the server 10 are received in the proper order by client computer 20, thus ensuring that the game state on client computer 20 is in synchronization with the known game state for the client computer on server 10. Messages sent from client computer 20 may be lost or unacknowledged, and the program operates to substitute a more updated message for any lost packets. Furthermore, client computer 20 does not operate on messages sent by itself without acknowledgement by server 10, thus further guaranteeing the synchronization. While the above has been described with the acknowledgement, retransmit request and serial numbers as part of a header of a message, this is not meant to be limiting in any way, and is specifically meant to include the acknowledgement, retransmit request and/or the serial number being sent as a message included in the packet.

Fig. 4 illustrates a high-level functional block diagram of a preferred embodiment operative in accordance with the present invention. The system comprises client computer 20, server 10 and data connection 30. Client computer 20 comprises application 40, packet assembler 70, message storage 80 and packet disassembler 120, and sending/receiving functionality 160. Packet disassembler 120 comprises disassembling functionality 170, storing functionality 180, examiner 190, marker 200, transferring functionality 210, calculator 220 and notifying functionality 230. Server 10

is connected by data connection 30 to client computer 10 at sending/receiving functionality 160. Application 40 is associated with packet assembler 70 and transferring functionality 220. Packet assembler 70 is associated with message storage 80, notifying functionality 220 and sending/receiving functionality 160. Message storage 80 is further associated with disassembler 120 and particularly disassembling functionality 170. Disassembling functionality 170 is associated with storing functionality 180 and sending/receiving functionality 160, and storing functionality 180 is further associated with examining functionality 190 and calculator 220. Examining functionality 190 is further associated with marking functionality 200, and marking functionality 200 is associated with transfer functionality 210. Calculator 220 is further associated with notifying functionality 230, and notifying functionality 230 is associated with packet assembler 70.

In operation the system of Fig. 4 operates in a manner similar to that described above in relation to Fig. 1, Fig. 2 and Fig. 3. Messages generated by application 40 are assembled into packets in packet assembler 70, stored in message storage 80, and sent by sending/receiving functionality 160. Server 10 sends packets comprising messages to client computer 20 over data connection 30, and the packets are received by sending/receiving functionality 160. The packets are transferred to disassembler 120, at disassembling functionality 170, which operates to disassemble the packet into messages and any header. The serial number found in the header is stored by storing functionality 180. The packet is examined by examination functionality 190 to determine if a message stored in message storage 80 is acknowledged, or if a retransmit request has been received for a message stored in message storage 80. Marking functionality 200 operates to mark the messages to indicate the results of the examination, and transfer functionality 200 transfers the marked messages to control program 40. Calculating functionality 220 calculates if the received packet has been received in sequential order, or if a packet is missing, and if a packet is missing notifying functionality 220 operates packet assembler 70 to generate a retransmit request, indicating the last properly received packet.

Fig. 5 illustrates a high level block diagram of a server 10 of Fig. 1 according to a preferred embodiment operative in accordance with the present invention. For clarity, server 10 will be described as having only a single communication link with a single

client computer 20, however it is to be understood that server 10 may run multiple links simultaneously, and communicate with multiple client computers 20 over multiple data connections 30. Server 10 comprises game control unit 250, and communication module 160. Communication module 160 comprises transmit message buffer 60, packet assembler 70, packet storage 260, transmit packet buffer 90, transmission agent 100, one end of data connection 30, receive packet buffer 110, packet disassembler 120, received message buffer 130 and received message number storage 270. Game control unit 250 is associated with the input of transmit message buffer 60 at the input to communication module 160, and transmit message buffer 60 is associated with an input of packet assembler 70. Packet assembler 60 is associated with an input of message storage 80 and with an input of packet transmit buffer 90. Packet transmit buffer 90 is associated with an input of transmission agent 100, and an output of transmission agent 100 is associated with the input of receive packet buffer 110. Transmission agent 100 is further bi-directionally associated with one end of data connection 30. The output of receive packet buffer 110 is associated with the input of packet disassembler 120, and an input of packet disassembler 120 is associated with the output of packet storage 260. One output of packet disassembler 120 is further associated with the input of received message buffer 130, and the output of received message buffer 130 is associated with the input of game control unit 250 across the output of communication module 160. Packet disassembler 120 is further bi-directionally associated with received message number storage 270, and the output of received message number storage 270 is associated with an input of packet assembler 70. A further output of packet disassembler 120 is further associated with an input of packet assembler 70.

In operation, game control unit 250 generates messages for client computer 20 (not shown). The messages may be placement of game elements of players, identification of game elements and players in a particular area of the game, updates regarding the game state, or other management messages. Messages generated by game control unit250 are stored in message transmit message storage 60, which in a preferred embodiment comprises a first in, first out (FIFO) memory identified with a specific client computer 20. Packet assembler 70 operates at regular intervals to collect messages from transmit message buffer 60, and to assemble them into packets. In one embodiment,

packet assembler 70 operates every 200 millisecond, and in a second embodiment packtet assembler 70 operates every 300 milliseconds. Packet assembler 70 further operates to sequentially number the packets being assembled. The packets are assembled with a header comprising the sequential number of the packet, the number of messages in the packet and the number of the last message received from the individual client computer 20 (not shown), which is being acknowledged, collected from received message number storage 270.

A copy of the packet is stored in packet storage 260, and the packet is sent to transmit packet buffer 90. Transmission agent 100 operates to transmit any packets found in transmit packet buffer 90 over data connection 30 to client computer 20 (not shown). In a preferred embodiment transmission agent 100 operates under the UDP/IP protocol.

Packets received by transmission agent 100 from client computer 20 (not shown) over data connection 30 are transferred to received packet buffer 110. Packet disassembler 120 operates to retrieve the received packet from received packet buffer 110 and disassemble the packet into messages destined for game control unit 250 and a header. The header comprises the sequential number of the last packet received by the client computer 20 (not shown) from server 10, or a request to retransmit any missing messages. In a preferred embodiment the header further comprises a sequential number of the first message in the packet and the total number of messages in the packet. The sequential number of the current message received from client computer 20 (not shown), and the total number of messages is stored in received message number storage 270. The sequential number of the last packet received by the client computer 20 is compared with the sequential number of the packets stored in packet storage 260, and the packet is then erased from packet storage 260. In the event that a retransmit request is received, the requested packets are retrieved from packet storage 260, and sent to packet assembler 70 for retransmittal. A feature of this invention is that messages from the server 10 are resent until confirmation of receipt occurs, whereas messages from client computer 20 are not retransmitted. Instead messages from client computer 20 when not acknowledge are regenerated by the application 40, and may thus be different than the original message which was not confirmed.

The serial number of the received messages are inspected to ensure that it is in sequence with the previously received messages, as stored in received message number storage 270. In the event that a message is missing, packet disassembler 120 notifies packet assembler 70 to generate a retransmit request. Packet assembler 70 utilizes the last message number in sequence found in message number storage 270, combined with the number of message associated with that last message in the retransmit request, thus indicated the last known proper received message. In the event that no message is missing, the messages are sent to received message buffer 130. Game control unit 250 polls received message buffer 130 to accept any new received messages.

Only a single game control unit 250 and a single set of associated communication modules are shown. It is to be understood that preferably for each client computer a separate logical set of communication modules is utilized, so that messages to and from each client computer are handled in accordance with the above protocol.

Fig. 6 illustrates a high level flow chart of the operation of disassembler 120 of Fig. 4. In step 2000, the disassembler receives a packet from receive packet buffer 110, and in step 2010 the serial number of the messages are identified and stored in received message number storage 270. In step 2020 the received packet is examined to determine if it comprises a retransmit request. In the event that it comprises a retransmit request, in step 2030 the stored packet is transferred from message storage 80 to packet assembler 70 with instructions to retransmit. It is to be understood that multiple packets may be stored and retransmitted. In a preferred embodiment when excessive numbers of unacknowledged packets are stored in packet storage 260, game control 250 assumes that client computer 20 is disconnected.

In the event that in step 2020 the received packet was determined not to comprise a retransmit request, or after step 2030, in step 2040 the serial number of the messages are checked against previously stored serial numbers in received message number storage 270, to determine if the packet has been received in sequence. In the event that a message is missing from the sequence, in step 2050 packet assembler 70 is notified to send a retransmit request to client computer 20, otherwise in step 2060 the packet is examined to find an acknowledgement. Preferably, the acknowledgment is part of the header. In the event that the received packet comprises an acknowledgement, in step

2070 the stored packet with matching sequential number is erased from message storage 80. In the event that in step 2060 no acknowledgement was found, or after step 2070, in step 2080 the messages are sent to game control unit 250 through received message buffer 130. Game control unit 250 thus receives messages which have been received in sequence, and will generate any required acknowledgements as further messages to client computer 20.

Fig. 7a and Fig. 7b illustrate a graphical representation of a reverse position structure (hereinafter "RevPos") on two different client computers 20 according to a preferred embodiment operative in accordance with the present invention. RevPos is a preferred embodiment in accordance with the present invention of a data structure comprising maps, elements and player information. In prior art systems each game element has associated with it a particular position on a map. The reverse positioning structure of the present embodiment allows the client computer 20 to identify for a given position the location and identification of all elements of all relevant players. In order to minimize bandwidth requirement the game area is divided into regions, preferably squares or hexagons. Each regions is classified as an inner region, border region or an outer region.

Fig. 7a illustrates the RevPos at a particular time for player 1, and Fig. 7b illustrates the RevPos at the same time, or game state, for player 2. For clarity the map is shown as a two dimensional map of equal squares, however it is to be understood that this is not meant to be limiting in any way and is intended to include other types of maps, including three dimensional maps and hexagonal regions. In Fig. 7a game pieces of player 1 are labeled A1, B1 indicating piece A and B respectively of player 1. Similarly the game pieces of player 2 which are relevant are labeled herein B2 and B2. Regions 1 to 8 are not within immediate reach of any of player 1's pieces, which are assumed to only interact within the current region or an adjacent region, and thus regions 1 to 8 are unmarked. Regions 9, 10, 11, 13, 17, 18 and 19 are each within immediate reach of piece A1 and regions 10, 11, 12, 14, 16, 18, 19 and 20 are within immediate reach of piece B1. These regions are not currently occupied by any piece of player 1, and are thus marked with a dotted pattern and are considered border regions. Regions 14 and 15 currently

contain piece A1 and B1 respectively, and are thus considered inner regions, and are marked with a hash pattern.

Application 40 of client computer 1 calculates interactions for all moves within inner regions 14 and 15, and the movement or action of all pieces of any other player within those inner regions is reported to client computer 20 of player 1 by game control unit 250 on server 10. Regions 9, 10, 11, 12, 13, 16, 17, 18, 19 and 20 are considered border regions for player 1, as they border on regions containing pieces for player 1. Server 10 notifies player 1 of all data regarding changes in the game state in border regions, and the result of any interactions with pieces in border regions is received from game control unit 250 on server 10.

Regions 1 to 8 are not adjacent to any pieces of player 1, and game control unit 250 does not informat application 40 on client computer 20 of player 1 of actions that occur in these regions. This conserves bandwidth, as only relevant information is being sent to player 1.

Fig. 7b illustrates the RevPos of player 2, in which pieces of player 2 are labeled A2, B2 and C2 indicating piece A, B and C respectively of player 2. Similarly the relevant game pieces of player 1 labeled A1 and B1 are shown. Regions 1, 5, 9 and 13 are not within immediate reach of any of player's 2 pieces, which are assumed to only interact within the current region or an adjacent region, and are therefore unmarked. Regions 2, 3, 4, 6, 8, 10, and 12 are each within immediate reach of piece A2, but unoccupied by any other piece of player 2, and are thus marked with a dotted pattern indicating a border region. Regions 6, 8, 10, 12, 14, and 16 are within immediate reach of piece B2 but unoccupied by any other piece of player 2, and are thus marked with a dotted pattern indicating a border region. Regions 10, 12, 14, 16,18, 19 and 20 are within immediate reach of piece C2 but unoccupied by any other piece of player 2, and are thus marked with a dotted pattern indicating a border region. Regions 7, 11 and 15 currently contain pieces A2, B2 and C2 respectively, and are thus considered inner regions and are marked with a hash pattern.

Application 40 of client computer 2 calculates interactions for all moves within inner regions 7, 11 and 15, and the movement or action of all pieces of any other player within those inner regions is reported to client computer 2 by game control unit 250 on

server 10. Regions 2, 3, 4, 6, 8, 10, 12, 14, 16, 18, 19 and 20 are considered border regions for player 2, as they border on regions containing pieces for player 2. Game control unit 250 notifies application 40 on computer 20 of player 2 of all data regarding changes in the game state in border regions of player 2, thus the result of any interactions with pieces in these regions is received from game control unit 250 on server 10.

Regions 1, 5, 9, 13 and 17 are not adjacent to any pieces of player 2, and game control unit 250 does not inform application program 40 on client computer 20 of player 2 of actions that occur in these regions. Not providing information about such outer regions conserves bandwidth, as only relevant information is being sent to player 2. It is important to note that thus player 2 received information on action of other pieces occurring in region 3, and can calculate results of actions in region 7, whereas player 1 receives no information about either region 3 or 7. This asymmetrical information sharing minimizes the required bandwidth.

In a preferred embodiment, messages regarding movement or action of all pieces within an inner region comprise player requested actions, and do not comprise a step by step calculation of the game state in those regions. Application 40 of players for which this region is an inner region calculate the game state for inner regions utilizing the broadcast messages of player requested actions, thus minimizing bandwidth. As indicated above, application 40 only acts upon its own messages upon acknowledgement, thus ensuring that all players are in synchronization with the game state on server 10 without requiring detailed messaging of game state updates for inner regions.

Fig. 8 illustrates a high level graphical representation of the structure of a single region in the RevPos on server 10, comprising player information 280 and game element information 290. Player information 280 comprises a list of all players who are sent information regarding movements in the region, which is all players to which this region is an inner or boundary region. Game element information 290 comprises all game elements within the region, with information regarding which player owns the elements. Each single regions RevPos is updated continuously on server 10 by game control unit 250 in response to moves made by each player and sent by each application 40.

Fig. 9 illustrates a high level block diagram of a system for a network-based game in which some of the active players are artificial playing entities, comprising computers

20 each comprising application 40, artificial playing entity 300 and communication modules 50, server 10 and data communication 30. First computer 20 comprises application 40 being operated by player 1 associated with first communication module 50 and first artificial playing entity 300, designated entity A associated with second communication module 50. Second computer 20 comprises application 40 being operated by player 2 associated with first communication module 50 and second artificial playing entity 300, designated entity B, associated with second communication module 50. Third computer 20 comprises application 40 being operated by player 3 associated with first communication module 50 and artificial playing entity 300, designated entity C associated with second communication module 50. First, second and third computer 20 are connected each connected by data communication lines 30 to server 10. In the event that player 1 wishes to cease playing, in a first embodiment player 1 passes control of all or some of player 1's playing pieces to player 2. This is accomplished by sending a message in the manner described above in relation to Fig. 3 and Fig. 4 to server 10 to transfer the appropriate pieces to player 2 by updating the server RevPos as discussed above in relation to Fig. 8.

In a second embodiment, upon first computer 20 disconnecting from the network, control of player 1's pieces are transferred to an artificial playing entity. Artificial playing entities are computer modules designed to play the pieces of the game according to a pre-programmed routine in the absence of a human player and may comprise sophisticated rule sets or artificial intelligence or like technologies. The artificial playing entity may be associated with the server, or preferably may be available amongst the computing resources found on other computers connected to the network. In a preferred embodiment server 10 searches for an available artificial playing agent from amongst the connected computing resources in a manner that will be described further herein.

The choice of playing agents A, B or C preferably is accomplished by game control unit 250. In one embodiment the choice is accomplished so as to minimize bandwidth at any one time. Thus if player 1 and player 2 share many inner and border regions, information being sent to application 40 of second computer 20 for player 2 may be utilized by second computer 20 to update entity B, artificial playing agent 260 on second computer 20. A further consideration utilized by game control unit 250 is

available computing resources on each client computer 20, as certain computers 20 may be have insufficient computing resources support one or more artificial playing agents 260, whereas other computers 20 may have sufficient computing resources to operate one or multiple artificial playing agents 260. In a preferred embodiment client computer 20 regularly updates game control unit 250 of available computing resources and available bandwidth. In a further preferred embodiment optimization to find the minimum bandwidth is not accomplished, but rather the first client computer 20 sharing inner regions with the disconnecting player and having sufficient resources is selected.

Fig. 10 illustrates a high level flow chart of the operation of game control unit 250 for the operation of finding an available artificial playing agent. In step 2200, game control unit 250 receives a request to continue the player's gameplay with an available artificial agent 300, and in step 2210 game control unit 250 searches RevPos information as described above in relation to Fig. 8 for a client computer sharing inner regions with the requesting player. In a preferred embodiment multiple client computers are ranked and the player closest to the requesting player in number of inner regions is chosen. In step 2220 the found client computer is checked to ensure sufficient available computing resources and bandwidth. If in step 2220 the first found client computer has sufficient resources and bandwidth, in step 2230 the artificial agent 300 is enabled on the chosen computer. In step 2240 the player information and game element information RevPos files are updated as discussed above in relation to Fig. 8. The selected client computer 20 regularly updates game control unit 250 about available resources, and if resources become unavailable, game control unit 250 will again transfer the requesting player to another available agent as indicated above.

Upon the player returning to the game, game control unit 250 transfers all of the players game elements from the current artificial agent 300 to the player, and disables the playing agent.

Fig. 11 illustrates a high level functional diagram of the operation of game control unit 250 for finding a close available computing resource and transferring control to an artificial agent 300. Game control unit 250 comprises receiver 320, searcher 330, bandwidth information storage 340, available resource storage 350, approver 360, updater 370, position maintainer 380, notifier 390, determiner 400 and divider 410. Receiver 320

is associated with searcher 330 and searcher 330 is bi-directionally associated with chooser 360. Approver 360 is associated with bandwidth information storage 340, available resource storage 350, and updater 370. Updater 370 is associated with maintainer 380, and maintainer 380 is associated with enabler 390 and determiner 400. Determiner 400 is associated with divider 410.

In operation, divider 410 divides the game space into regions, which as indicated above are preferably squares or hexagons, and determiner 400 determines for each region which players have the region as an inner, border or outer region. Maintainer 380 maintains the list of regions during game play, and notifier 390 operates to notify players of changes in the game state in their inner and border regions.

Receiver 320 receives a request from a player to be replaced by an artificial agent 300, and operates searcher 330 to find available computing resources for artificial agent 300. Preferably, the available computing resource comprises a player which shares at least one inner region with the requesting player. Still further preferably the searcher finds the available computing resource which shares the maximal amount of inner regions with the requesting player. Approver 360 checks available bandwidth stored in bandwidth information storage 340 and available computing resources stored on available resource storage 350 and if the choice has the needed resources the approved choice is passed to updater 370. If the choice is does not have the needed resource, searcher 330 is prompted to make another choice. Updater 370 updates the lists maintained in maintainer 360 with the location of the playing agent, so that control of the game pieces and subsequent tracking is properly accomplished.

It is appreciated that certain features of the invention, which are, for clarity, described in the context of separate embodiments, may also be provided in combination in a single embodiment. Conversely, various features of the invention which are, for brevity, described in the context of a single embodiment, may also be provided separately or in any suitable subcombination.

Unless otherwise defined, all technical and scientific terms used herein have the same meanings as are commonly understood by one of ordinary skill in the art to which this invention belongs. Although methods similar or equivalent to those described herein

can be used in the practice or testing of the present invention, suitable methods are described herein.

All publications, patent applications, patents, and other references mentioned herein are incorporated by reference in their entirety. In case of conflict, the patent specification, including definitions, will prevail. In addition, the materials, methods, and examples are illustrative only and not intended to be limiting.

It will be appreciated by persons skilled in the art that the present invention is not limited to what has been particularly shown and described hereinabove. Rather the scope of the present invention is defined by the appended claims and includes both combinations and subcombinations of the various features described hereinabove as well as variations and modifications thereof which would occur to persons skilled in the art upon reading the foregoing description.

WHAT IS CLAIMED IS:

1. A method for network gaming involving a virtual game area divisible into regions, and a plurality of networked players each having at least one playing piece, comprising:

   determining for each region and for each player whether said region is an inner region in which said player has a game piece, a border region which is within access of a player's piece or an outer region which is not accessible in this move by the player;

   maintaining a relationship of players and game elements, said relationship maintaining whether said region is an inner region or a border region for each player; and

   notifying only those players for which the region is a border region of changes in the game state.

2. The method of claim 1 further comprising notifying only those players for which the region is an inner region of all player actions in said inner region.

3. The method of claim 1 further comprising notifying only those players for which the region is an inner region or a border region of all player actions in said regions.

4. The method of claim 1 wherein said regions are squares.

5. The method of claim 1 wherein said regions are hexagons.

6. The method of claim 1 further providing game persistency by:

receiving notification by a server of an impending disconnection of a player;

searching available computing resources to determine an available substitute playing agent;

choosing said available substitute playing agent; and

updating said maintained relationship said substitute playing agent.

7. The method of claim 6 wherein said searching further comprises finding said available substitute agent having inner regions in common with said disconnecting player.

8. The method of claim 6 wherein said searching further comprises finding said available substitute agent having the maximum amount of inner regions in common with said disconnecting player

9. The method of claim 6 wherein said available substitute agent is an artificial agent.

10. A system for network gaming involving a virtual game area divisible into regions comprising:

a plurality of networked players, each player having at least on playing piece;

a server, connected to each playing entity by a data connection, said server providing:

determining functionality for determining for each region and for each player whether said region is an inner region in which said player has a game piece, a

border region which is within access of a players piece or an outer region which is not accessible in this move by the player;

maintaining functionality associated with said determining functionality for maintaining in said server a relationship between players and game elements, said relationship comprising whether said region is an inner region or a border region for each player; and

notifying functionality associated with said maintaining functionality for notifying only those players for which the region is a border region of changes in the game state.

11. The system of claim 10 wherein said notifying functionality is further for notifying only those players for which the region is an inner region of player actions in said inner region.

12. The system of claim 10 wherein said notifying functionality is further for notifying only those players for which the region is an inner region or a border region of player actions in said regions.

13. The system of claim 10 wherein said regions are squares.

14. The system of claim 10 wherein said regions are hexagons.

15. The system of claim 10 wherein said server further provides:

receiving functionality for receiving notification by said server of an impending disconnection of a player;

searching functionality associated with said receiving functionality for an available substitute playing agent;

bandwidth availability functionality for determining the currently available bandwidth to each available substitute playing agent;

computing resource availability functionality for determining the currently available computing resources of each available substitute playing agent;

choosing functionality associated with said searching functionality said bandwidth availability functionality and said computing resource availability functionality for choosing said available substitute playing agent having required bandwidth requirements and available computing resources; and

updating functionality associated with said choosing functionality and said maintaining functionality for updating said maintaining functionality with said substitute available playing agent.

16. The system of claim 15 wherein said choosing functionality is further for finding said available substitute agent having inner regions in common with said disconnecting player.

17. The system of claim 15 wherein said choosing functionality is further for finding said available substitute agent having the maximum amount of inner regions in common with said disconnecting player.

18. The system of claim 15 wherein said available playing agent is an artificial agent.

19. A method for persistent network gaming for a plurality of networked players, comprising;

receiving notification by a server of an impending disconnection of a networked player, said player having at least one gaming piece;

searching available computing resources to determine an available substitute playing agent;

choosing said available substitute playing agent; and

transferring control of said gaming pieces to said available agent.

20. The method of claim 19 wherein said searching further comprises finding said available substitute agent having minimum bandwidth requirements.

21. The method of claim 19 wherein said available substitute agent is an artificial agent.

22. The method of claim 19 wherein said wherein at least some of said available computing resources are co-resident with at least one networked player.

23. A system for persistent network gaming comprising:

a plurality of networked players; and

a server, said server providing:

(i) receiving functionality for receiving notification by said server of an impending disconnection of a player;

(ii) searching functionality associated with said receiving functionality for searching available computing resources to determine an available substitute playing agent;

(iii) bandwidth determining functionality associated with said searching functionality for determining the bandwidth requirements and availability for each available substitute playing agent;

(iv) choosing functionality associated with said ranking functionality for choosing said available substitute playing agent with minimum bandwidth requirements; and

(v) transferring functionality associated with said choosing functionality for transferring said at least one game pieces to said chosen available substitute playing agent.

24. The system of claim 23 wherein said available playing agent is an artificial agent.

25. The system of claim 23 wherein at least some of said additional computing resources are co-resident with at least one networked player.

26. A system for network based gaming comprising:

a client computer; and

a data connection connecting said client computer to a server;

said client computer comprising an application program and a communication module, said communication module further comprising:

a) assembling functionality for assembling messages from said application program into packets for delivery to said server computer over said data connection;

b) message storage functionality associated with said assembling functionality for storing a copy of said messages prior to said delivery;

c) received packet storage number functionality for maintaining a list of received packets from said server;

d) disassembler functionality associated with said received packet storage number functionality for disassembling said received packets from said server, said disassembler functionality further providing:

i) examining functionality associated with said message storage functionality for examining said received packet for an acknowledgement or retransmit request of said stored message by said server;

ii) marking functionality associated with said message storage functionality and said examination functionality for marking said stored messages with said acknowledgement or said retransmit request ; and

iii) transfer functionality associated with said marking functionality for transferring said marked messages from said message storage to said application program.

27. The system according to claim 26 wherein said disassembler functionality further provides calculating functionality associated with said storage functionality for determining if said received packet is in numerical sequence with previously received packets.

28. The system according to claim 26 wherein said disassembler functionality further provides notifying functionality associated with said calculating functionality for generating a retransmit request to said server.

29. A method for network gaming comprising:

assembling messages from an application program into a packet for delivery to a server computer over a data connection;

storing a copy of said messages;

disassembling a received packet from said server computer, said disassembling comprising:

i) storing a serial number of said received packet;

ii) examining said received packet for successful acknowledgement or retransmit request of at least one of said stored messages;

iii) marking the messages of said stored messages with the result of said examination; and

iv) transferring said stored messages from said message storage to said application program.

30. The method according to claim 29 further comprising calculating if said received packet is in numerical sequence with previously received packets.

31. The method according to claim 30 further comprising generating a retransmit request in an out of sequence event.

32. The method according to claim 29 further comprising said application program acting upon said acknowledged messages, and generating new messages in response to said unacknowledged messages.

Fig. 1

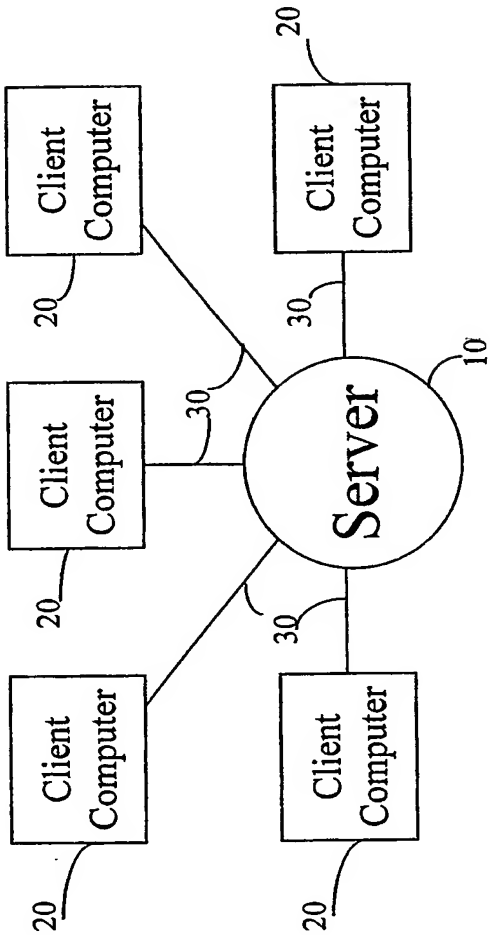
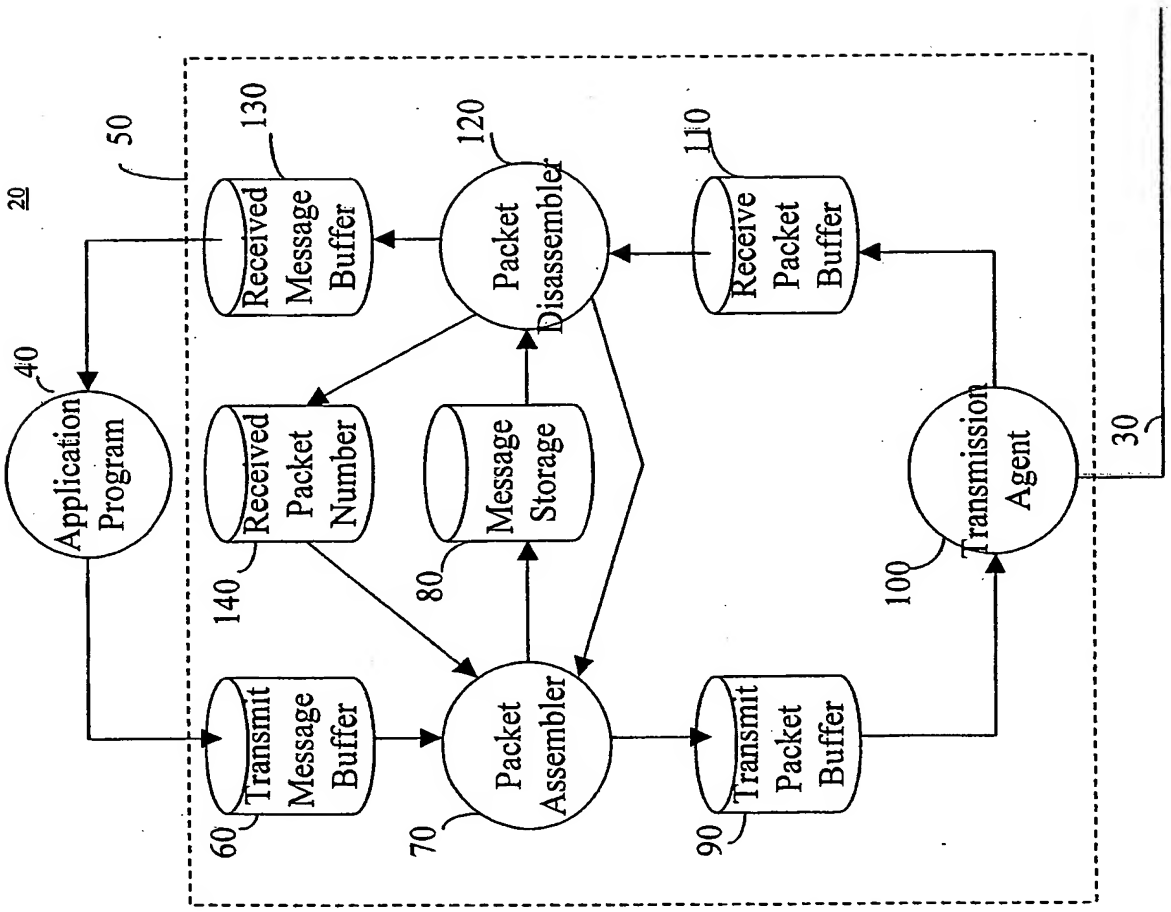

Client Computer
Client Computer
Client Computer
Client Computer
Client Computer
Server
10
20
30

Fig. 2



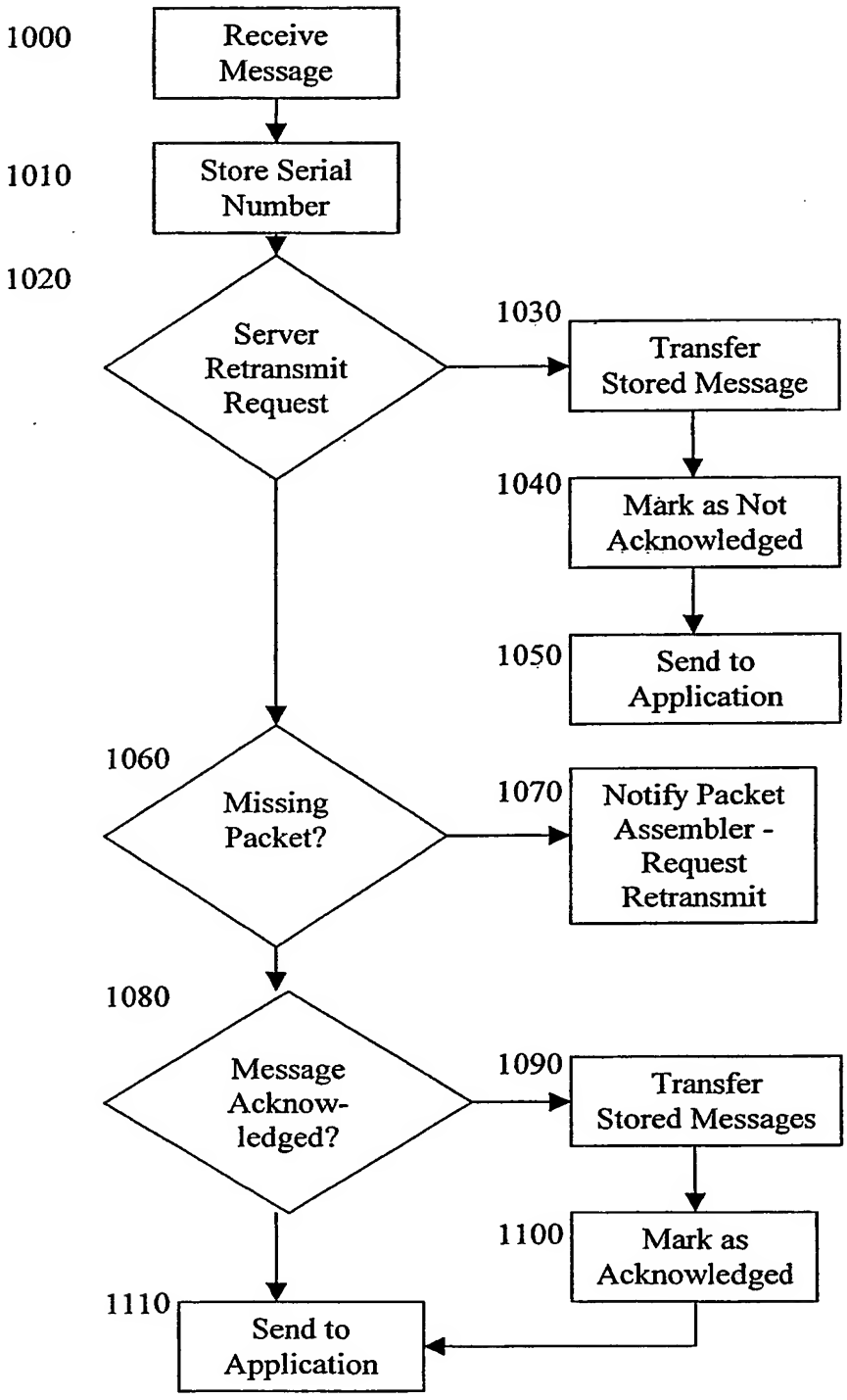


Fig. 3

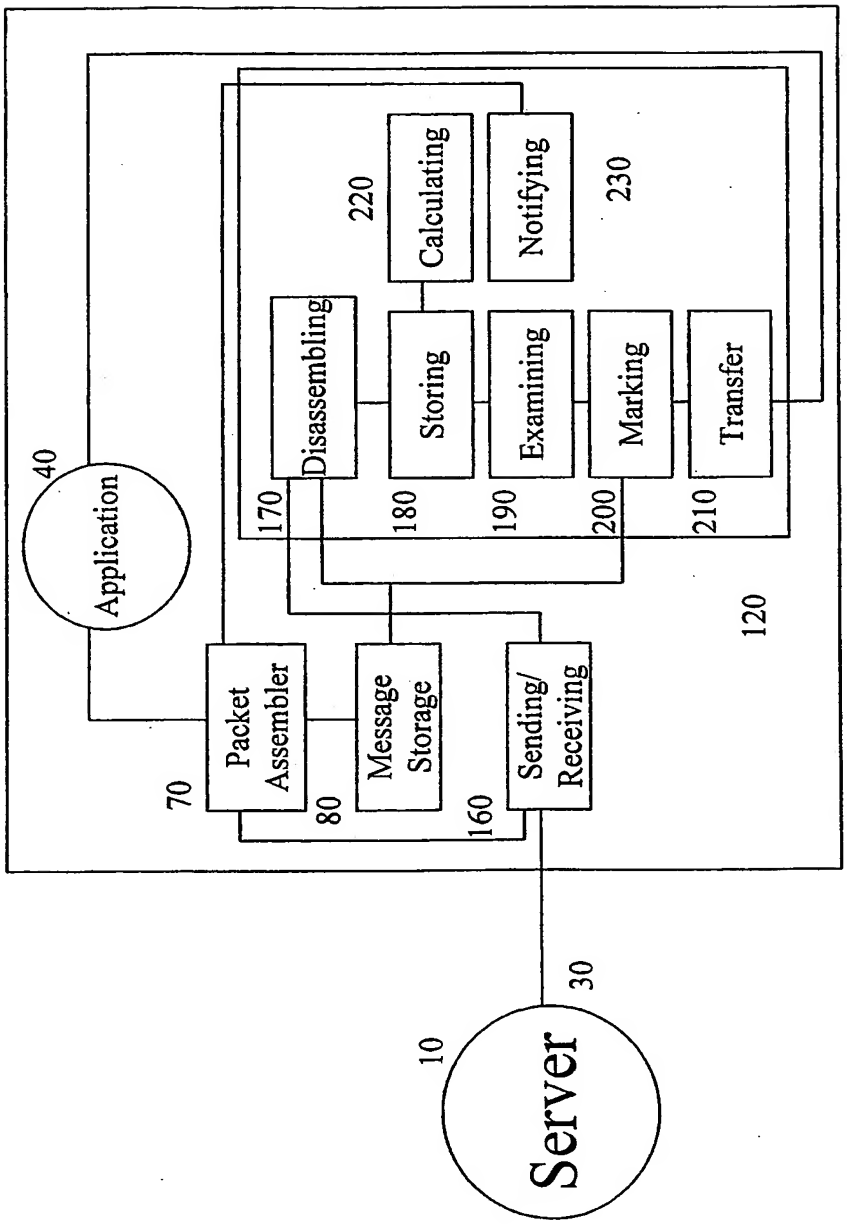
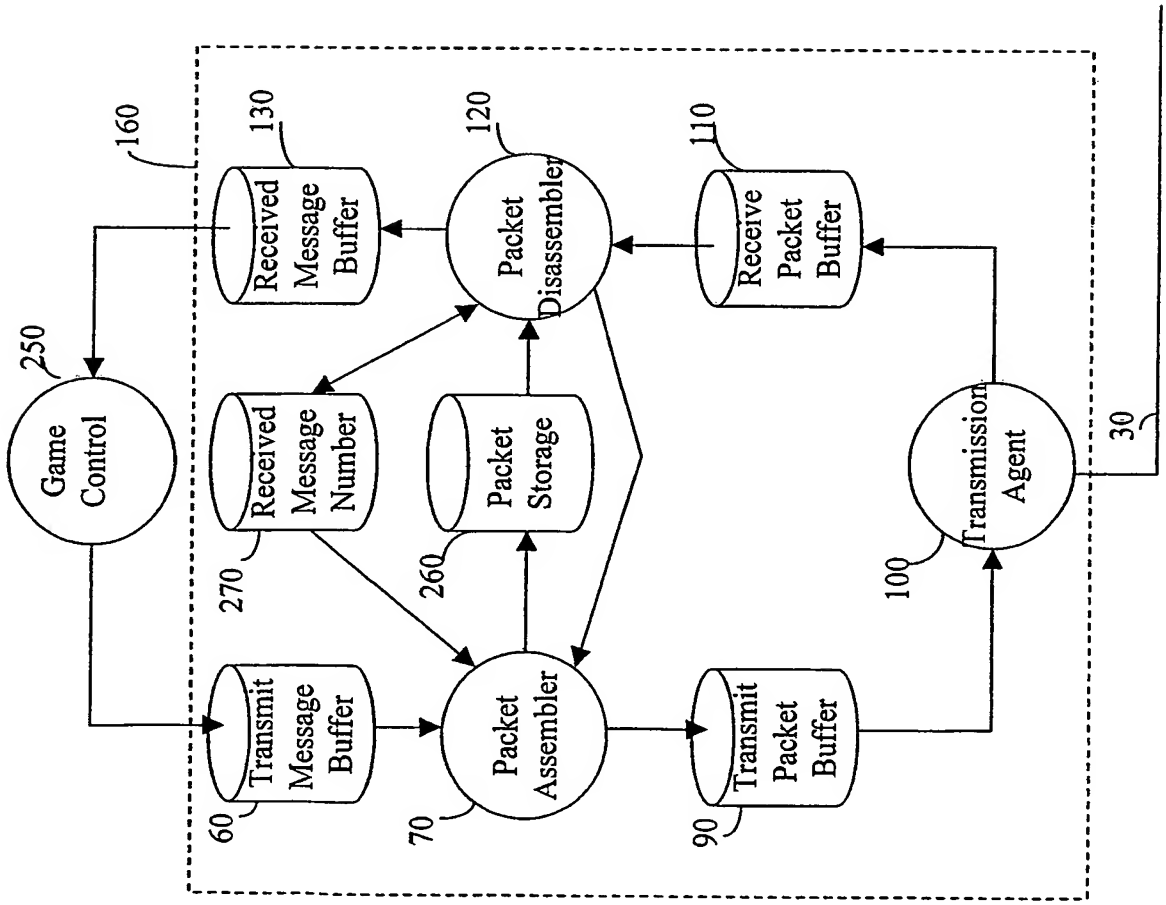

20
Application
40
Packet Assembler
70
Message Storage
80
Sending/ Receiving
160
Disassembling
170
Storing
180
Examining
190
Marking
200
Transfer
210
Calculating
220
Notifying
230
120
Server
10
30

Fig. 4

Fig. 5



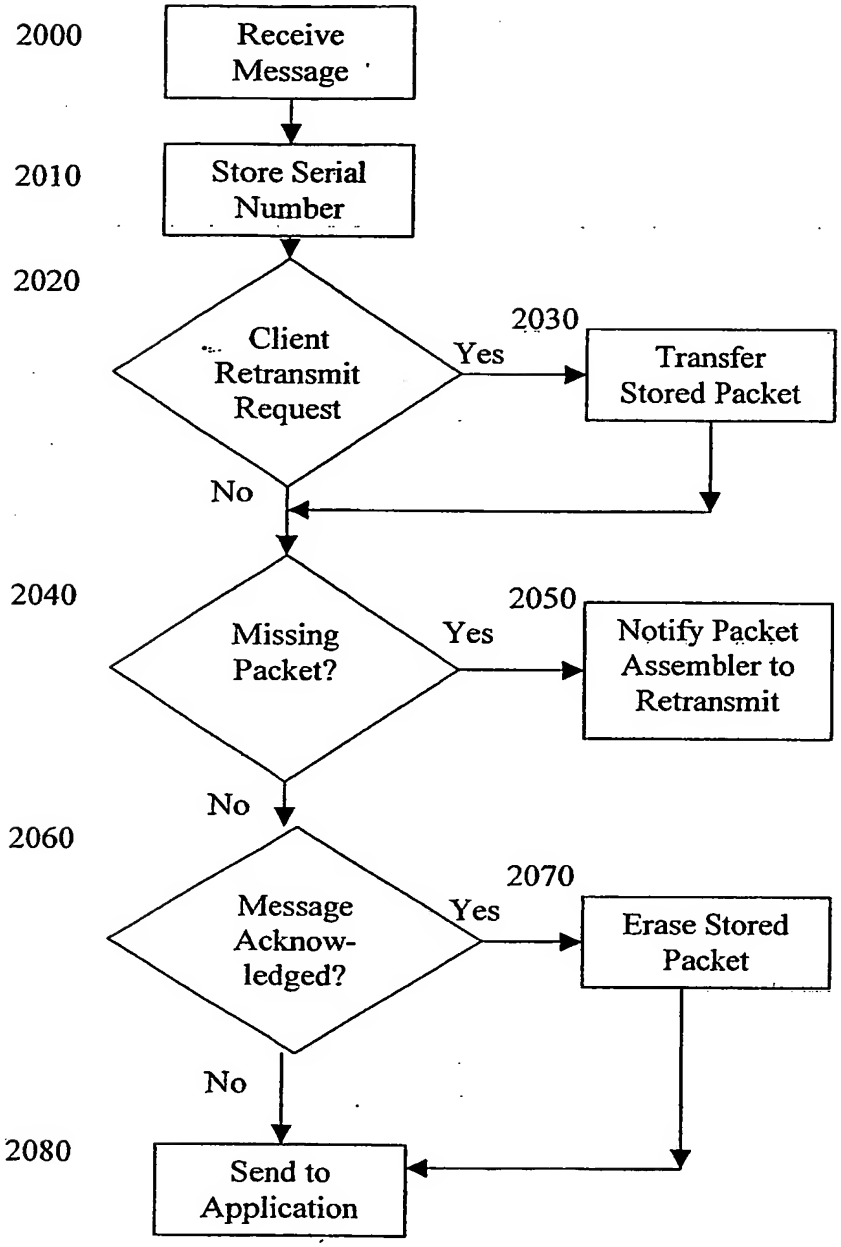

2000
Receive Message
2010
Store Serial Number
2020
Client Retransmit Request
Yes
2030
Transfer Stored Packet
No
2040
Missing Packet?
Yes
2050
Notify Packet Assembler to Retransmit
No
2060
Message Acknow-ledged?
Yes
2070
Erase Stored Packet
No
2080
Send to Application

Fig. 6

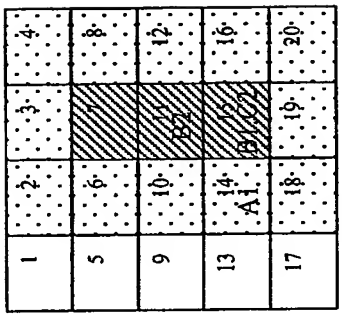


Fig. 7b

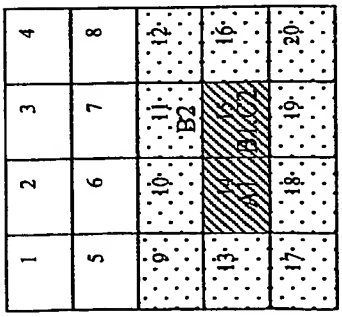


Fig. 7a

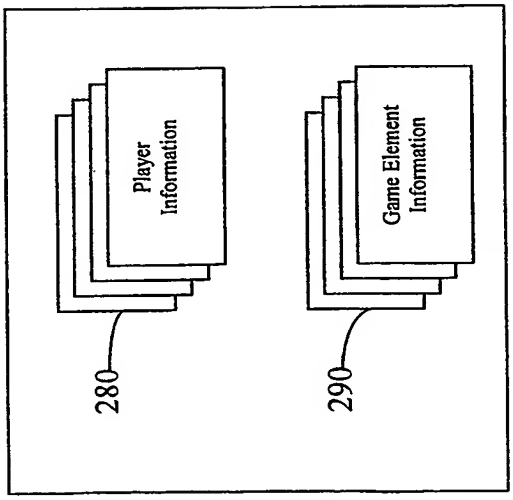
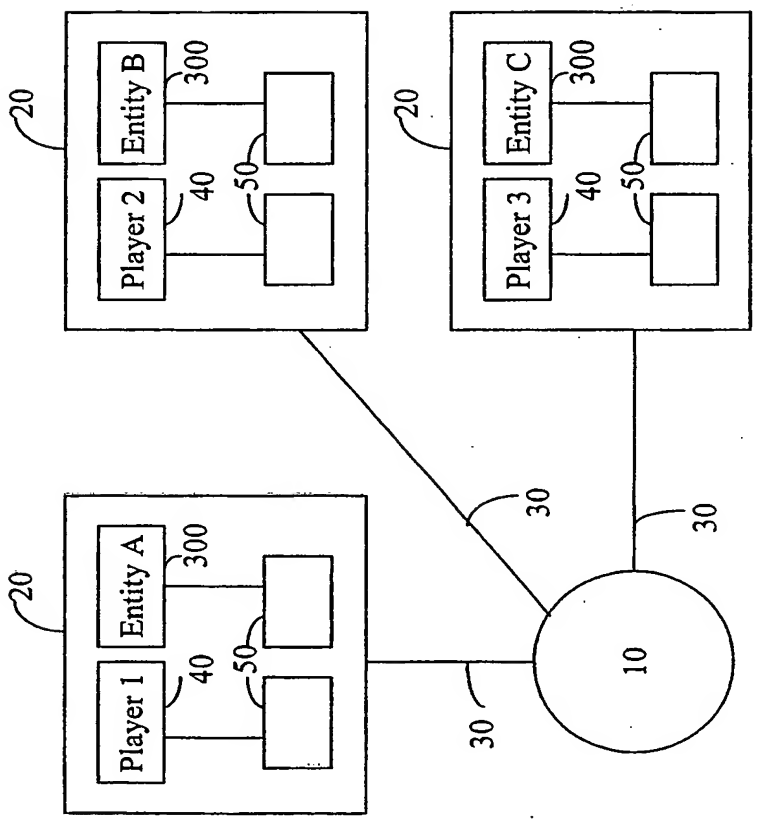


Fig. 8

Fig. 9



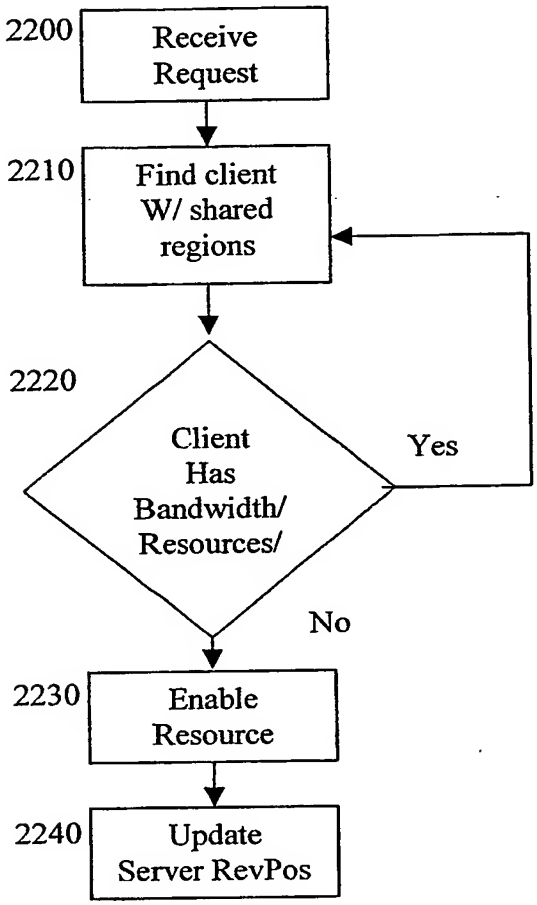
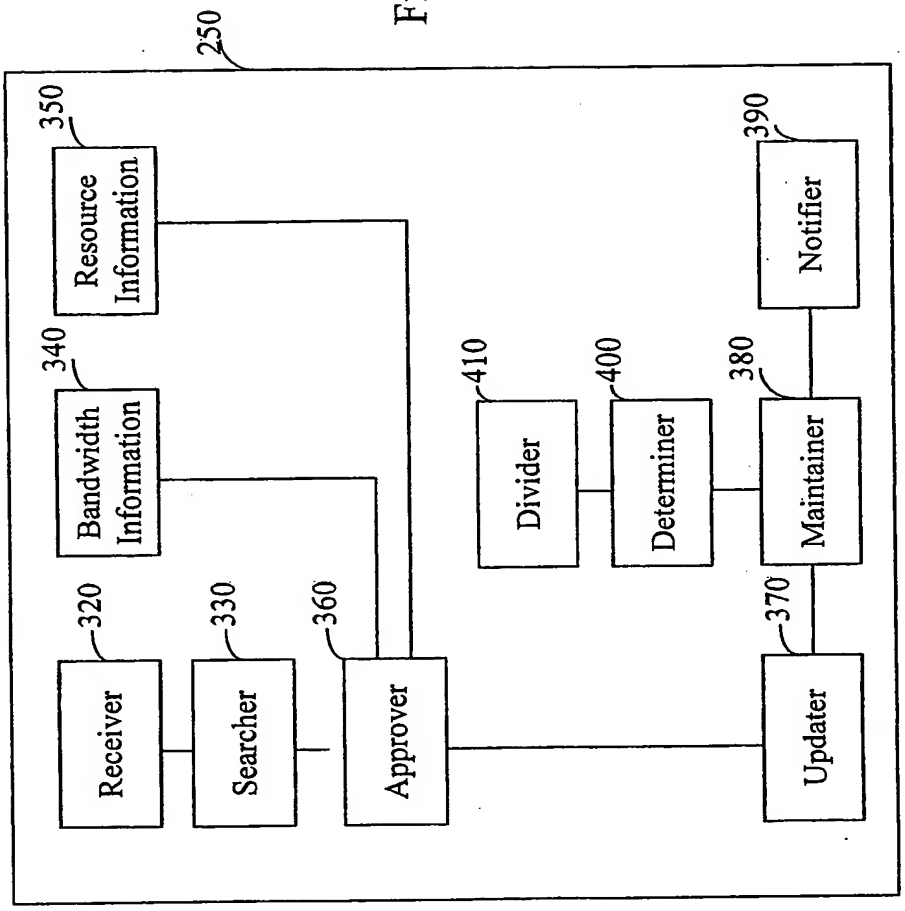

2200
Receive Request
2210
Find client W/ shared regions
2220
Client Has Bandwidth/ Resources/
Yes
No
2230
Enable Resource
2240
Update Server RevPos

Fig. 10

Fig. 11



INTERNATIONAL SEARCH REPORT

International application No.
PCT/IL02/01000

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : A63F 9/24
US CL : 463/42

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 463/42, 1, 30, 31, 37, 40, 41, 43

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Please See Continuation Sheet

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A,P	US 6,409,602 B1 (WILTSHIRE et al) 25 June 2002 (25.06.2002), see entire document.	1-32
A,P	US 6,406,371 B1 (BABA et al) 18 June 2002 (18.06.2002), see entire document.	1-32

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
04 April 2003 (04.04.2003)

Date of mailing of the international search report
29 APR 2003

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703)305-3230

Authorized officer
Kim Nguyen
Telephone No. (703)308-1078

Sheila H. Verlay
Paralegal Specialist
Tech. Center 3700

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

PCT/IL02/01000

Continuation of B. FIELDS SEARCHED Item 3:
EAST
search terms: network, virtual game, server, client computer, player